P-22

Performance Study of a Data Flow Architecture

George Adams

Research Institute for
Advanced Computer Science

RIACS

Outline

- Motivation
- Methodology
- Data Flow
- Results

Extended Abstract

In September 1984 RIACS conducted a two-week study of the proposed MIT static data flow machine for applications of interest to NASA Ames and DARPA. NASA and RIACS scientists formed seven one- or two-person teams to study data flow concepts, the static data flow machine architecture, and the VAL language. Each team mapped its application onto the machine and coded it in VAL.

The application areas were computational fluid dynamics, computational chemistry, galactic simulation, linear systems, queueing network models, and artificial intelligence. The considerations for mapping these applications onto the machine were primarily architectural: the number of individual processing elements (PE), the size of the instruction memory in each PE, the speed of the PEs, the instruction issue rate, the size of the routing network among the PEs, and the size and speed of the array memory. The goal in mapping was to maximize the number of busy PEs and to minimize the traffic on the routing network. The target machine contained 256 PEs and was capable of an aggregate rate of 1.28 GFLOPS.

The principal findings of the study were:

1. Five of the seven applications used the full power of the target machine – they sustained rates of 1.28 GFLOPS. The galactic simulation and multigrid fluid flow teams found that a significantly smaller version of the machine (16 PEs) would suffice.
2. A number of machine design parameters including PE function unit numbers, array memory size and bandwidth, and routing network capability were found to be crucial for optimal machine performance. Thus, studies of this type can provide valuable feedback to machine architects.
3. The study participants readily acquired VAL programming skills. A very high level programming environment is essential to make the data flow machine usable by most programming scientists, however, because of the complexity of the machine architecture. For example, tools to aid debugging and mapping VAL programs onto the architecture are required.
4. We learned that application-based performance evaluation is a sound method of evaluating new computer architectures, even those that are not fully specified. During the course of the study we developed models for using computers to solve numerical problems and for evaluating new architectures. We feel these models form a fundamental basis for future evaluation studies.

RIACS

Institute of the Universities Space Research Association (USRA)

- Began operation June 1983

Purpose

- Strengthen CS at Ames Research Center
- Strengthen ties to University and Industry

Technical Program

- Independent research (core)
- Joint projects (tasks)

Central Theme –

Integration of concurrent processing and artificial intelligence into every aspect of scientific investigation

Study of MIT Static Data Flow Machine and VAL Language

- Jointly funded by NASA and DARPA
- Focussed on algorithms for several applications

Goals

- Gain experience with VAL, pipe–structured programming, machine architecture
- Estimate machine performance
- Evaluate system usability
- Assess study value and methodology

Motivation

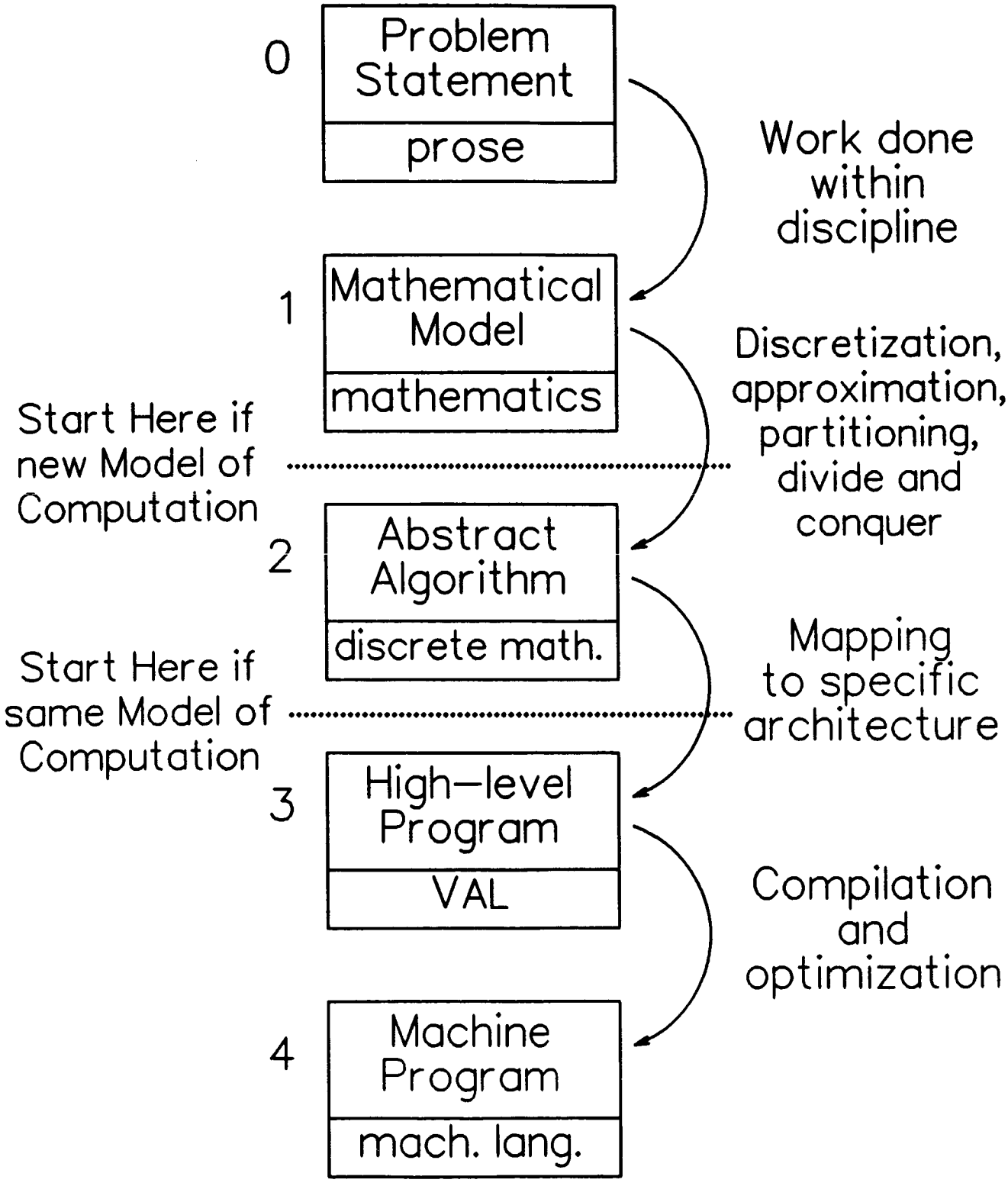
- Progress in concurrent processing research limited by lack of understanding of interactions between
  - computational models
  - system architecture
  - classes of algorithms
- Goal is **usable systems**
- Model of computation can be an experiment focus

Nature of Study

- Six application areas studied

  Computational fluid dynamics
  Computational chemistry
  Galactic simulation
  Linear systems
  Natural language processing
  Queueing network models

- Eleven team members – expert in discipline, novice to data flow
- Two weeks release time from normal duties
- Study held away from daily distractions
- Familiar computer environment ported to RIACS facility for each participant; no time spent learning new system



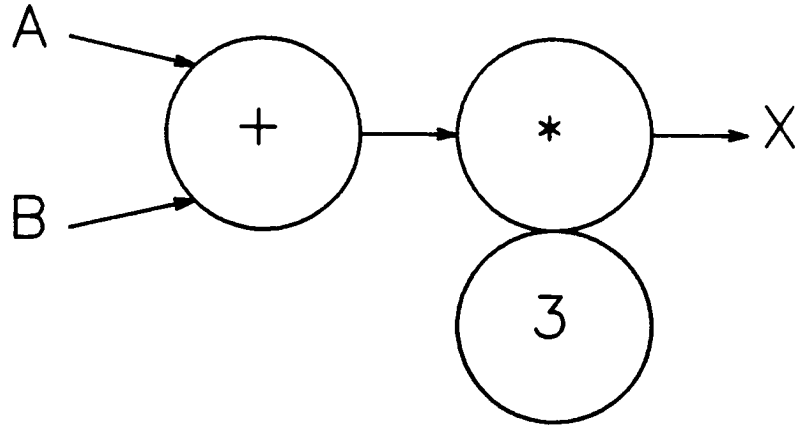
Steps in Problem Solving

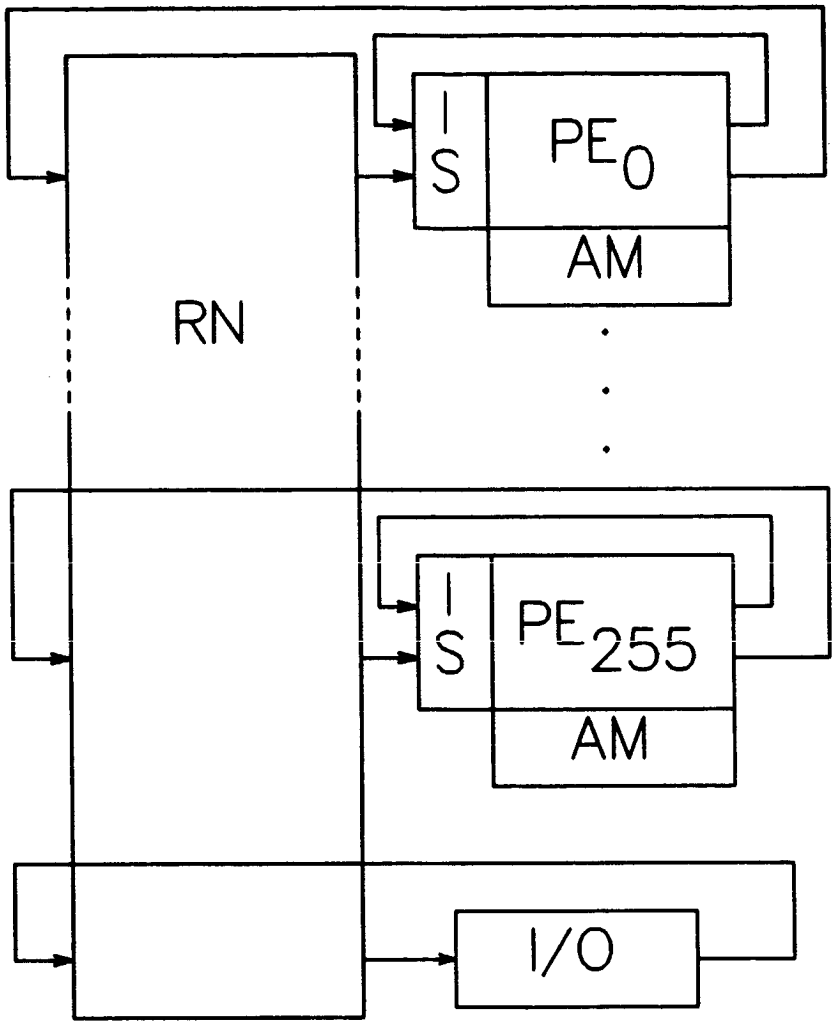
Static Data Flow Model of Computation

- Program is a directed graph
  - graph nodes represent instructions
  - operands travel on edges
- Instruction **enabled** by presence of operands and permission to send result

Example:

$$X = 3*(A+B)$$





RN – Routing Network.
PE – Processing Elements.
IS – Instruction Store.
AM – Array Memory.
I/O – Input/Output.

Static Data Flow Machine Architecture

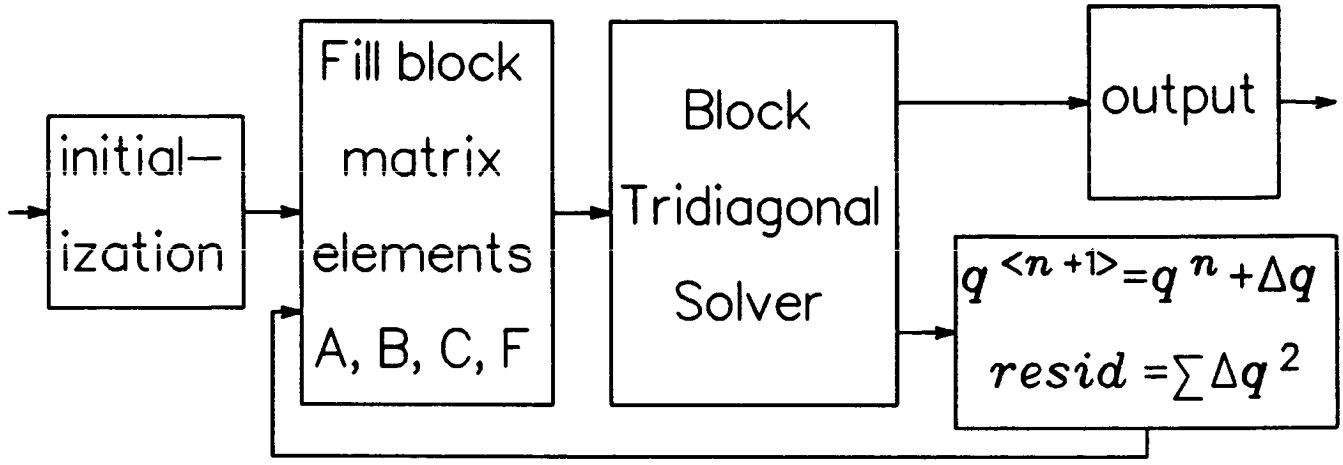
VAL – Value-Oriented Algorithmic Language

- Functional
- Single assignment
- Side-effect free
- Modular – external and internal functions
- Data treated as values not objects

```
A, B, C :=
    forall J in [1,N]
        X: real := square-root (real(J));
        construct J, X, X/2
    endall
```

Computational Fluid Dynamics

CSM



Structure of CSCM Data Flow Program

- 3–D with 10,000 grid points per 4–PE "cluster" to achieve peak speed
- PE instruction store probably too small
- Recomputation done to save memory

Computational Fluid Dynamics

Euler Method

- Method routinely used by industry
- Describes inviscid, compressible fluid flow

Findings

- Handle boundary conditions better than vector processors
- Limited parallelism; machine with 16 PEs may be good choice
- Tiny scalar speed of single PE may be problem

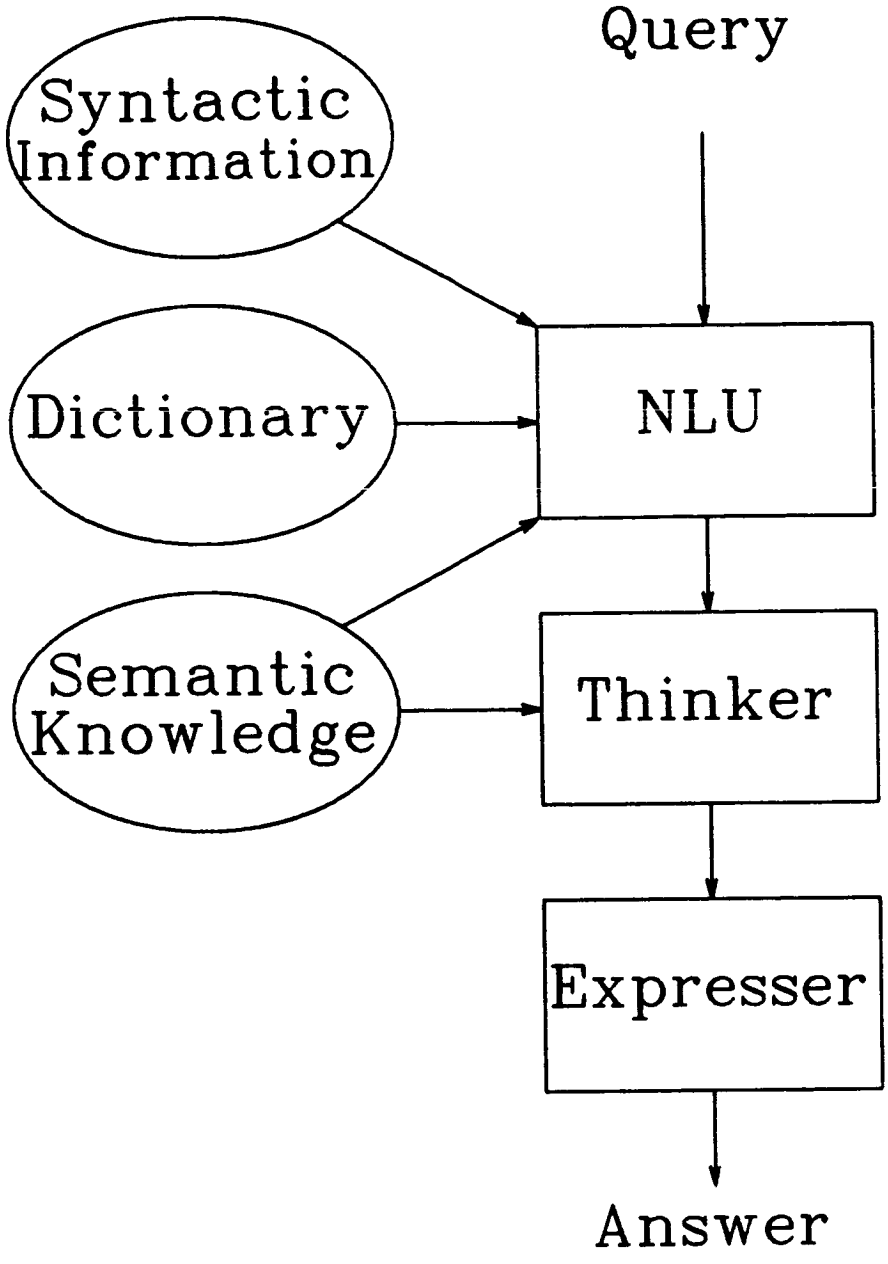
Computational Chemistry

- Calculate material properties from fundamental physics
- Complements and supplements experiment
- Problem sizes often
  - 40,000×40,000 random sparse arrays
  - 400 billion floating point operations
  - 64 Mword files
- Currently studying Kapton to understand degradation experienced on Shuttle missions

Findings

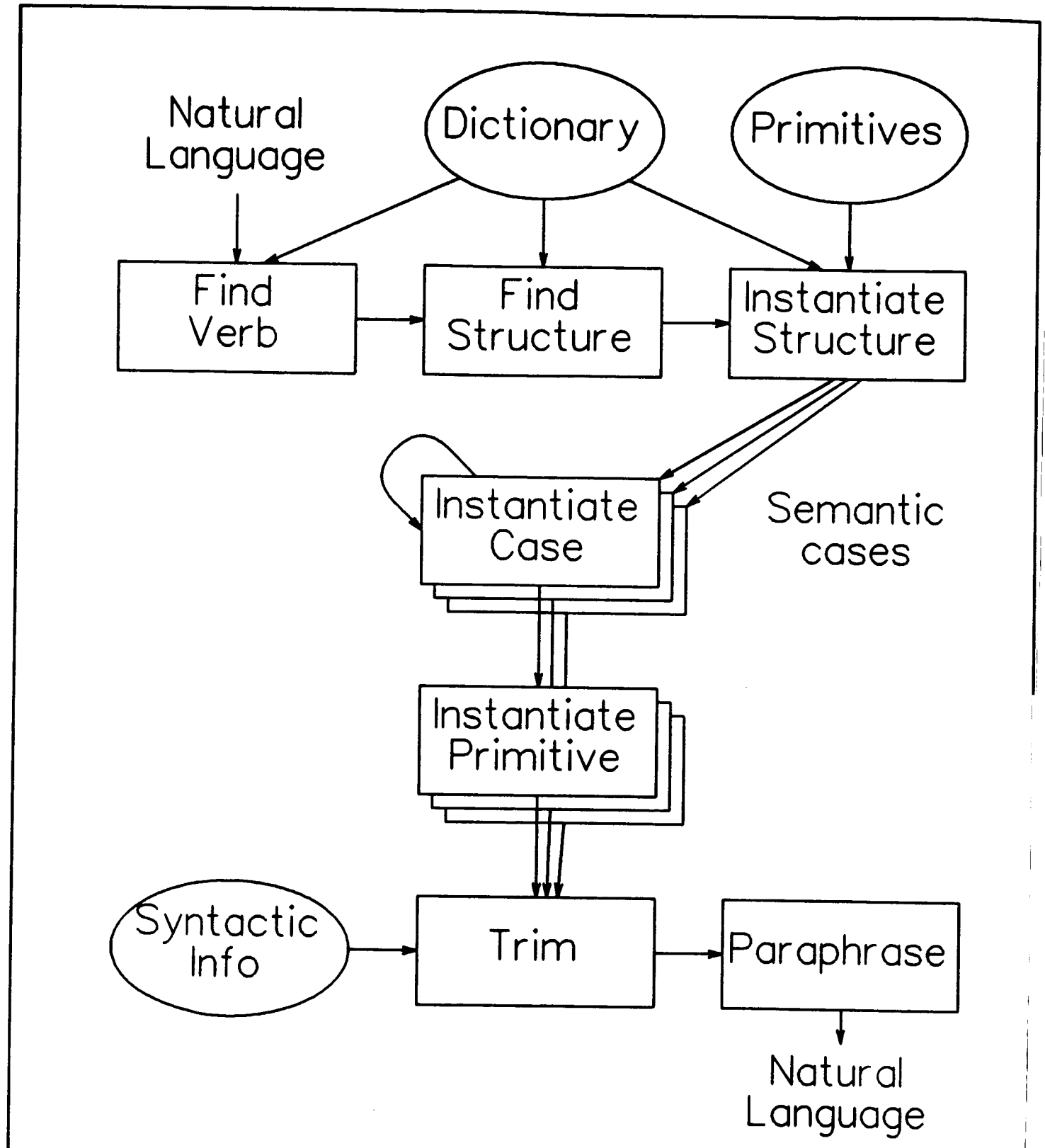
- Algorithms adaptable
  - Diatomic Integral Code (DERIC) may operate at near 1.2 GFLOP rate
- Major problem was I/O
  - inadequate bandwidth
  - data set partitioning
- 2K instruction cell memory too small
- VAL compiler directives may be difficult to write
- Since machine is single user I/O wait time cannot go to another job
- Fewer, higher–performance PEs recommended

Natural Language Processing



NLU – Natural Language Understander

Structure of Natural Language Processor



Structure of the NLU VAL Program

Findings

- Lack of global databases a significant problem
- Lack of high–level support for I/O
- New algorithmic approaches must be devised that readily exhibit parallelism

Queueing Network Models

- A tool for deriving performance estimates for computer systems and communication networks
- Compute performance measures such as throughput, response time, utilization, average queue length

Findings

- Practical considerations limit useful problem sizes to those that can be solved interactively on sequential computers
- Approximate techniques suffice for very large problems
- The most parallel algorithm did not lead to efficient parallel code due to fan–in/fan–out restrictions

What We Learned

- VAL readily learned
- Data flow model of computation readily learned
- Most applications could use all parallelism available
- Smaller machines may be quite cost effective

- VAL programming environment must improve; debugging support
- Efficient sparse matrix handling may require hardware support
- Array memory bandwidth should be increased
- Array memory size should be increased
- Bandwidth for disk I/O should be increased
- Appropriate number of PE logic units difficult to assess

Benefits of Study

- Broken some of the resistance of users to look at new computer architectures
  - strong effort made by team members
- Computer scientists working with users
  - machine and language design changes
- Knowledge on how to conduct future studies
  - other combinations of model/machine/applications

References

1. G. B. Adams, R. L. Brown, and P. J. Denning, *Report on an Evaluation Study of Data Flow Computation*, RIACS Technical Report, TR 85.2, April 1985.

2. G. B. Adams, R. L. Brown, and P. J. Denning, *On Evaluating Concurrent Architectures*, RIACS Technical Report, May 1985.

3. G. B. Adams and P. J. Denning, *A Ten Year Plan for Concurrent Processing Research*, RIACS Technical Report, March 1984.

4. W. B. Ackerman and J. B. Dennis, *VAL -- A Value-Oriented Algorithmic Language: Preliminary Reference Manual*, MIT/LCS/TR-218, Massachusetts Institute for Technology, June 1979.

5. J. B. Dennis, "Data Flow Ideas for Supercomputers," *Proc. COMPCON84*, February 1984.

6. J. B. Dennis, G. R. Gao, and K. W. Todd, "Modeling the Weather with a Data Flow Computer," *IEEE Transactions on Computers*, vol. C-33, July 1984.

7. P. J. Denning and K. C. Sevcik, "Execution of Queueing Network Analysis Algorithms on a Data Flow Computer Architecture," RIACS Technical Report, 1985.

8. D. A. Reed and M. L. Patrick, "Iterative Solution of Large, Sparse Linear Systems on a Static Data Flow Architecture: Performance Studies," RIACS Technical Report, February 1985.

9. M. L. Merriam, "Application of Data Flow Concepts to a Multigrid Solver for the Euler Equations," *Proc. 2nd Copper Mountain Conf. on Multigrid Methods*, April 1985.

10. E. Levin, *Performance Evaluation of a Static Data Flow Processor for Transformations of Large Arrays*, RIACS Technical Report, TR 85.1, Jan. 1985.

11. D. S. Eberhardt and K. Rowley, *An Analysis of Static Data Flow to a Sample CFD Algorithm*, RIACS Technical Report, Mar. 1985.